МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
"МИРЭА - Российский технологический университет"

РТУ МИРЭА

Институт Информационных Технологий

Кафедра Вычислительной Техники

ПРАКТИЧЕСКАЯ РАБОТА

по дисциплине

«Системный анализ данных СППР»

Студент группы:ИКБО-42-23 Голев С.С.
(Ф. И.О.студента)

Преподаватель Железняк Л.М.
(Ф.И.О. преподавателя)

Москва 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Генетический алгоритм представляет собой один из наиболее известных и эффективных эвристических методов для решения задач оптимизации. Его основная идея основана на моделировании процессов естественного отбора и эволюции, происходящих в биологических популяциях. Подобно тому как в природе особи с более высокой приспособленностью имеют больше шансов на выживание и воспроизводство, в генетическом алгоритме решения с лучшими характеристиками получают преимущество в процессе формирования новых поколений.

В работе рассматривается применение генетического алгоритма для нахождения минимума функции. Такой подход позволяет получать приближённые оптимальные решения в задачах, где пространство поиска велико и содержит множество локальных экстремумов. Важным компонентом метода является использование операторов селекции, кроссовера и мутации, которые обеспечивают эффективный баланс между наследованием удачных свойств и исследованием новых областей пространства решений.

Благодаря своей универсальности и способности находить хорошие решения даже в условиях сложного рельефа целевой функции, генетический алгоритм широко применяется в задачах глобальной оптимизации. Он позволяет уменьшить вероятность преждевременной сходимости и способствует нахождению более качественных решений по сравнению с классическими градиентными методами.

7 ГЕНЕТИЧЕСКИЙ АЛГОРИТМ

Генетический алгоритм является эволюционным алгоритмом и полностью основан на итерации родительских точек, что делает его крайне эффективным в множестве задач.

7.1 Цель и задачи практической работы

Целью практической работы является освоение генетического метода и его применение для решения задач оптимизации различного типа. В рамках работы необходимо изучить принципы работы алгоритма.

Для достижения поставленной цели необходимо выполнить следующие задачи:

1. Реализовать генетический алгоритм для нахождения минимума заданной функции;
2. Проанализировать влияние параметров алгоритма на качество и скорость сходимости;
3. Сравнить полученные результаты с другими методами;
4. Сделать выводы о применимости генетического алгоритма для задач оптимизации.

7.2 Постановка задачи

В рамках практической работы необходимо реализовать нахождение минимума функции с помощью генетического алгоритма.

Как тестовая функция была выбрана функция Била.

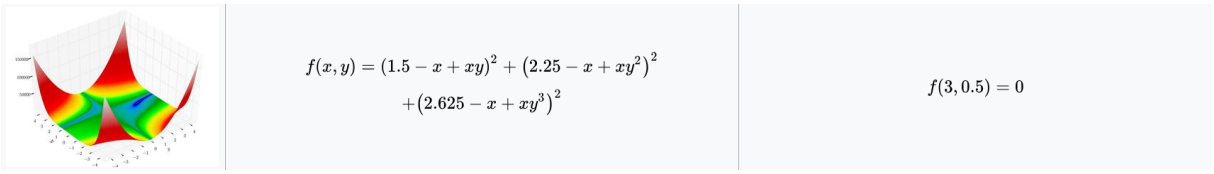

Рисунок 7.1 – Функция Била

7.3 Ручной расчёт

Выполним ручной расчёт одной итерации, для функции Била диапазон значений для *x* и *y* равен [–4.5; 4.5].

Для расчёта будут использованы следующие формулы:

Фитнес функция:

$$fit(x) = \frac{1}{1+f(x)} \tag{7.1}$$

Функция для линейной комбинации родителей при создании нового потомка:

$$c = \alpha * p_1 + (1 - \alpha) * p_2 \tag{7.2}$$

Где p – координаты точек родителей и α – случайный параметр:

Формула расчёта мутации:

$$x_i^{t+1} = x_i^t + \delta \tag{7.3}$$

Где δ – случайная величина распределённая по нормальному закону. Данная формула работает как для первой так и для второй координаты.

Начнём расчёт, для начала инициализируем популяцию, назначив каждой особи координаты и для этих координат вычеслив функцию и фитнесс функцию по формуле 7.1:

Особь 1 : x = 2.1, y = -1.8, f(x,y) = 204.95, fit = 0.005;

Особь 2 : x = 0.3, y = 3.2, f(x,y) = 177.64, fit = 0.006;

Особь 3 : x = -2.7, y = -0.5, f(x,y) = 81.14, fit = 0.012;

Особь 4 : x = 1.5, y = 2.8, f(x,y) = 1333.75, fit = 0.0007;

Особь 5 : x = -3.9, y = 1.2, f(x,y) = 0.85, fit = 0.540;

Особь 6 : x = 4.2, y = -2.3, f(x,y) = 3339.16, fit = 0.0003;

Особь 7 : x = -0.8, y = 0.9, f(x,y) = 16.34, fit = 0.058;

Особь 8 : x = 3.6, y = -3.1, f(x,y) = 12993.33, fit = 0.00008;

Особь 9 : x = -1.4, y = 4.1, f(x,y) = 8955.14, fit = 0.00011;

Особь 10 : x = 2.8, y = 1.5, f(x,y) = 127.49, fit = 0.0077;

Первое поколение сформировано, теперь на его основе сформируем новое поколение, сначала производится турнирный отбор, где выбираются 3 случайные особи, и потом выбирается одна лучшая особь из этих трёх.

Родитель 1:

Пусть случайные особи: 2, 5, 9, из данных особей лучшей является особь под номером 5.

Родитель 2:

Пусть случайные особи: 3, 7, 10, из данных особей лучшей является особь под номером 7.

Следующий этап, после выбора родителей, формирование детей, с вероятностью 80% два ребёнка образуются с помощью формулы 7.2 и с вероятностью 20% оба ребёнка копируют родителей. Воспользуемся формулой 7.2 с учётом что $\alpha = 0.4$.

Ребёнок 1: x = -2.04, y = 1.02;

Ребёнок 1: x = -2.66, y = 1.08;

Произведём мутацию детей, основываясь на формуле 7.3, мутация производится с вероятностью 0.1, для каждой координаты отдельно.

Допустим что для ребёнка 1 для координаты x мутация не прошла, а для координаты y прошла, тогда высчитав мутацию по формуле 7.3 новые значения координат: x = -2.04 и y = 1.09.

Допустим что для ребёнка 2 для координаты x мутация прошла, а для координаты y не прошла, тогда высчитав мутацию по формуле 7.3 новые значения координат: x = -2.69 и y = 1.08.

Так мы повторяем этот алгоритм, пока у нас не образуется 10 детей, которые будут является вторым поколением, полностью основанном на первом поколении. После чего производится расчёт нового поколения и так заданное количество итераций.

7.4 Результат работы генетического алгоритма

Реализуем нахождение минимума функции с помощью генетического алгоритма, возьмём 50 особей и 300 итераций. Реализация представлена в приложении Б.

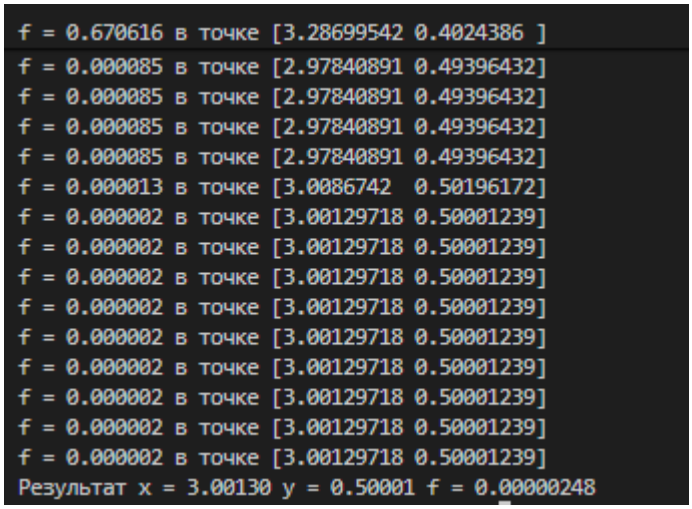

Рисунок 7.2 – Пример нахождение минимума генетическим алгоритмом

7.5 Результат работы нахождения минимума

В ходе практической работы был выполнен ручной расчёт одной итерации генетического алгоритма.

Данный расчёт наглядно показывает принцип работы метода: формирование нового поколения на основе работы старого и небольшой мутации что позволяет не только эффективно находить минимум функции, но и рассматривать окрестности точек, благодаря небольшой мутации.

Далее была выполнена кодовая реализация алгоритма. В результате работы показано, что алгоритм являются достаточно эффективными инструментом для задач оптимизации.

ЗАКЛЮЧЕНИЕ

В результате выполнения работы были реализован алгоритмы нахождения минимума функции. Проведённые эксперименты показали, что генетический алгоритм способен успешно находить решения, близкие к оптимальным, даже для сложных задач. Благодаря основе на предыдущих поколениях и наличии мутации алгоритм устойчив к локальным минимумам и легко позволяет находить минимум функции за небольшое количество итераций и вычислений.

СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. Python Software Foundation. Python Documentation — [Электронный ресурс]. URL: https://docs.python.org/3/ (дата обращения: 15.09.2025).

2. Лутц М. Изучаем Python. 5-е изд. / пер. с англ. — Санкт-Петербург: Символ-Плюс, 2019. — 1648 с.

3. Баляев С. А. Объектно-ориентированное программирование. Учебное пособие. — Москва : ФОРУМ, ИНФРА-М, 2020. — 256 с.

4. Гринберг Д. Программирование на Python 3. Подробное руководство. — Москва : Вильямс, 2014. — 832 с.

ПРИЛОЖЕНИЯ

Приложение Ж – Код программы “Генетический алгоритм”

Приложение Ж

Код Генетического алгоритма

Листинг Ж.1 — Основной алгоритм программы

```
import numpy as np

def f(x, y):
    return (1.5 - x + x*y)**2 + (2.25 - x + x*(y**2))**2 + (2.625 -
x + x*(y**3))**2


def genetic_minimize_beale(
        pop_size=50,
        n_generations=200,
        x_bounds=(-4.5, 4.5),
        y_bounds=(-4.5, 4.5),
        pc=0.8,
        pm=0.1,
        mutation_sigma=0.1,
        tournament_size=3,
        seed=123
    ):
    rng = np.random.default_rng(seed)

    pop = np.empty((pop_size, 2))
    pop[:, 0] = rng.uniform(x_bounds[0], x_bounds[1],
size=pop_size)  # x
    pop[:, 1] = rng.uniform(y_bounds[0], y_bounds[1],
size=pop_size)  # y

    def fitness(ind):
        x, y = ind
        value = f(x, y)
        return 1.0 / (1.0 + value)
    def evaluate_population(pop):
        return np.array([fitness(ind) for ind in pop])
    def tournament_select(pop, fit):
        idx = rng.integers(0, len(pop), size=tournament_size)
        best_i = idx[np.argmax(fit[idx])]
        return pop[best_i].copy()
    def crossover(parent1, parent2):
        if rng.random() < pc:
            alpha = rng.random()
            child1 = alpha * parent1 + (1 - alpha) * parent2
            child2 = alpha * parent2 + (1 - alpha) * parent1
        else:
            child1, child2 = parent1.copy(), parent2.copy()
        return child1, child2
```

Листинг Ж.2 — Продолжение листинга Ж.1

```
    def mutate(ind):
        for i in range(len(ind)):
            if rng.random() < pm:
                ind[i] += rng.normal(0.0, mutation_sigma)
        ind[0] = np.clip(ind[0], x_bounds[0], x_bounds[1])
        ind[1] = np.clip(ind[1], y_bounds[0], y_bounds[1])
        return ind
    best_ind = None
    best_fit = -np.inf
    for gen in range(n_generations):
        fit = evaluate_population(pop)

        gen_best_i = np.argmax(fit)
        if fit[gen_best_i] > best_fit:
            best_fit = fit[gen_best_i]
            best_ind = pop[gen_best_i].copy()

        print(f"f = {f(best_ind[0], best_ind[1]):.6f} в точке
{best_ind}")

        new_pop = []

        while len(new_pop) < pop_size:
            p1 = tournament_select(pop, fit)
            p2 = tournament_select(pop, fit)

            c1, c2 = crossover(p1, p2)

            c1 = mutate(c1)
            c2 = mutate(c2)

            new_pop.append(c1)
            if len(new_pop) < pop_size:
                new_pop.append(c2)

        pop = np.array(new_pop)

    best_x, best_y = best_ind
    best_value = f(best_x, best_y)
    return best_x, best_y, best_value


if __name__ == "__main__":
    x_opt, y_opt, f_opt = genetic_minimize_beale(
        pop_size=80,
        n_generations=300
    )
    print(f"Результат x = {x_opt:.5f} y = {y_opt:.5f} f =
{f_opt:.8f}")
```